Por outro lado, se considerarmos o mesmo número de batatas para serem descascadas o texto pode levar a resultados diferentes, supondo que o ato de descascar as batatas vai escurecendo a cor da saia. Estando as batatas mais sujas, a saia vai deixar de ficar clara mais rapidamente, resultando em menos batatas descascadas.

Um algoritmo deve ser *determinístico*, isto é, dadas as mesmas condições iniciais deve produzir, depois de executado, os mesmos resultados.

Só estamos interessados nos algoritmos que terminam em um tempo finito.

1.2 PROGRAMAS E ESTRUTURAS DE DADOS

Vamos agora ao computador: trata-se de uma máquina capaz de seguir uma certa espécie de algoritmos, chamados *programas*. Ele possui uma memória, capaz de armazenar dados, e uma unidade aritmética, que é capaz de causar mudanças nos dados armazenados na memória. Além disso, é capaz de comunicar-se com o mundo exterior através de operações de entrada e saída.

É da existência de computadores que vem o nosso interesse por algoritmos, mas, infelizmente, um computador real só é capaz de seguir programas que estejam escritos em linguagens de máquina, que normalmente é obscura e desconfortável.

Dentre todas as objeções a se usar linguagem de máquina para fazer um programa gostaríamos de destacar uma: um texto escrito em linguagem de máquina é uma seqüência monótona de instruções, não refletindo, portanto, a natureza do comportamento dinâmico de sua execução.

Estamos então diante da segunda situação: temos um computador real A, que só aceita linguagem de máquina, e queremos um computador ideal B, que aceita uma linguagem com construções que nos permitam exprimir de uma forma "natural" o nosso raciocínio algorítmico.

Fazemos então, de uma vez por todas, um programa (em linguagens de máquina) que faça o computador A se comportar como o computador B, e chamamos a este programa *compilador*.

Uma vez pronto o compilador, torna-se uma questão de ponto de vista dizer que estamos programando o computador A ou B, já que o conjunto computador + compilador (*hardware + software*) aceita a linguagem do computador B. É bem verdade que são necessárias duas fases para a execução de um programa, mas, em princípio, não faz mal a ninguém ignorar isto.

Como já dissemos, um algoritmo tenta estabelecer um efeito desejado no estado de um sistema, e o estado de um computador é determinado pela situação de sua memória em um dado momento. Ora, quando fazemos um programa, queremos algo de útil, e portanto queremos representar na memória uma parte do mundo real em que estamos interessados.

A pergunta imediata é: o *que* vamos representar na memória de um computador? Uma segunda pergunta é: *como* representar alguma coisa no computador? Antes de responder a estas perguntas, vamos nos permitir uma ligeira digressão sobre representações.

A base de todo progresso científico é a capacidade de, ao observar dois ou mais objetos, esquecer as suas possíveis diferenças aparentes e concentrar em algumas de suas semelhanças. Isto representa um processo de *abstração*, e o próximo passo é a *representação* desta abstração. Um exemplo de abastração são os números naturais — dados os mais diversos conjuntos dos mais variados objetos, todos somos capazes de identificar uma propriedade comum a todos, tal como possuir (digamos) três objetos.

Para a representação do número natural três, podemos usar, por exemplo, as representações abaixo:

3 (sistema decimal)
| | | (barrinhas)

Qual delas é a melhor? Ora, se representamos uma propriedade do mundo real, é porque estamos interessados em fazer alguma *manipulação* sobre as representações que corresponda a algum fenômeno do mundo real. Além disso, é óbvio que as operações feitas sobre as representações devem envolver um esforço mínimo comparado com uma manipulação dos objetos reais. Por exemplo, se quisermos saber quantas laranjas resultam da mistura de um saco com três laranjas com outro com quatro laranjas, sem fazer a mistura efetiva e contar, podemos fazer a operação de soma sobre as representações:

$$3 + 4 = 7$$
| | | + | | | | = | | | | | | |

Até este ponto, a representação por barrinhas apresenta uma indiscutível superioridade sobre a representação decimal, pois sua regra de soma é simplíssima, ao passo que para somar 3 e 4 (em decimal) precisamos saber de cor o resultado.

A situação se inverte por completo se necessitarmos também das operações de multiplicação e divisão, ou mesmo da soma de números grandes, ocasiões em que a notação posicional revela as suas grandes qualidades.

Um fato implícito nesta discussão é que, aos resultados obtidos através da manipulação das representações, devem corresponder resultados no mundo real, e, antes de aceitarmos qualquer par representação-manipulação, é necessário provar que existe esta correspondência, confrontando com a experiência em alguns casos.

Temos então neste processo três fases razoavelmente definidas:

a) A escolha das propriedades relevantes do objeto real que queremos representar.

b) A escolha da representação, com base nas operações a serem efetuadas sobre elas.

c) A demonstração de que o par representação-manipulação produz resultados de acordo com os fenômenos do mundo real correspondente.

Uma linguagem de programação é uma técnica de notação para programar, com a intenção de servir de veículo para: